International Journal of Computer Applications (0975 - 8887)
Volume 154 - No.1, November 2016

Data Pre-processing for a Neural Network Trained by an Improved Particle Swarm Optimization Algorithm

Tuan Linh Dang
Kochi University of Technology
Tosayamada, Kami City
Kochi 782-8502, JAPAN

Thang Cao
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku
Tokyo 113-8656, JAPAN

Yukinobu Hoshino
Kochi University of Technology
Tosayamada, Kami City
Kochi 782-8502, JAPAN

ABSTRACT

This paper proposes an improved version of particle swarm optimization (PSO) algorithm for the training of a neural network (NN). An architecture for the NN trained by PSO (standard PSO, improved PSO) is also introduced. This architecture has a data pre-processing mechanism which consists of a normalization module and a data-shuffling module. Experimental results showed that the NN trained by improved PSO (IPSO) achieved better performance than both the NN trained by standard PSO and the NN trained by back-propagation (BP) algorithm. The effectiveness concerning the recognition rate and the minimum learning error of the data pre-processing modules (normalization module, data-shuffling module) was also demonstrated through the experiments.

General Terms

Neural network, Particle swarm optimization, Data pre-processing

Keywords

Normalization, Data shuffling, Neural network, Particle swarm optimization, C language

1. INTRODUCTION

Nowadays, a classification system becomes an attractive target of research. The core of the classification system is a machine learning algorithm. One of the most widely used machine learning algorithms is a neural network (NN) [1–3].
To have the ability to solve a classification problem, the NN needs to be trained. Researchers have used the NN trained by particle swarm optimization (PSO) algorithm to overcome the limitation of NN trained by back-propagation (BP) algorithm in terms of the recognition rate and the convergence speed [4–7].
The classification tasks of the NN trained by PSO (NN-PSO) have been investigated in previous studies [7–10]. However, these studies used only standard PSO that may easily stick to a local minimum. It is beneficial to have a mechanism to keep the particle out of the local minimum in PSO algorithm.
The range of data attributes in a real-world application is diverse. There are the attributes that have higher values than other attributes. On the other hand, an activation function of the NN has an efficiency range. If input values in the training phase are too big, the activation function always fires. The NN becomes unable to learn [1, 11]. It needs to normalize the input data in the efficiency range of the NN in a pre-processing module of the NN-PSO system. Another aspect is also important for the pre-processing is the order of training data of the NN. The data-shuffling mechanism may prevent the situation of highly correlated training data when the unique information presented by each input data is small. In addition, the NN may learn faster with unpredicted data. If the NN learns all data from class one, it has the optimized weights for this class. In this situation, this NN needs to forget the class one before learning class two. The data-shuffling method may generate the training data that is not familiar with the NN [1, 11]. The data-shuffling module is also necessary for the pre-processing of the NN-PSO system.
The main contribution of this research is to propose an improved version of the PSO algorithm (IPSO). An architecture for the NN trained by PSO (standard PSO, IPSO) is also presented. This architecture has the pre-processing modules that contain the normalization module and the data-shuffling module. The comparison among three algorithms (NN trained by IPSO, NN trained by standard PSO, and NN trained by BP) is investigated in this paper. The operation of the standardization and the data-shuffling is also evaluated. These experiments were conducted with several public recognized databases.
This paper is presented as follows. Section 2 describes the related work about the NN and the PSO algorithm. Section 3 introduces the proposed IPSO algorithm. This section also presents the proposed design for the NN-PSO system with data pre-processing modules. Section 4 discusses the experiments and analyzes the results. Section 5 concludes this paper.

2. RELATED WORK

2.1 Neural network

The development of the NN is inspired by a human brain. Typically, The NN contains three classes of layer called the input layer, the hidden layer, and the output layer, respectively. The NN has only one input layer and one output layer, but it may have more than one hidden layers. Each layer has several nodes. The input data are transmitted from a node in one layer to the nodes in the next layer through an activation function. In this activation function, if the input of the node is high enough, the output of this node becomes

one. The sigmoid function has been widely used as the activation function as can be seen in equation 1 [1–3].

$$S(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

where x is the input data, $S(x)$ is the output of the activation function.

2.2 Particle swarm optimization algorithm

The PSO algorithm comes from social behaviors. When one particle in the swarm found an optimal position, other members will follow this particle. The calculation of the new position of each particle is based on the new velocity. The velocity update function is expressed in equation 2, and the new position update function is shown in equation 3. The new velocity of each particle depends on the current velocity, the best position of this particle, and the best position of all particles [12, 13].

$$\begin{aligned} v_p^{t+1} = w v_p^t + c_1 r_1 (x\_pbest_p^t - x_p^t) \\ + c_2 r_2 (x\_gbest^t - x_p^t) \end{aligned} \tag{2}$$

$$x_p^{t+1} = x_p^t + v_p^{t+1} \tag{3}$$

where v_p^{t+1}, x_p^{t+1} denote the velocity and the position of particle p at time $(t+1)$, c_1 and c_2 are coefficients, r_1 and r_2 are random numbers.

2.3 Linearly decreasing inertia weight

The linearly decreasing inertia weight was presented to improve the performance of the standard PSO algorithm by using a strategy for the weight control. This algorithm reduces the weights by iterations as can be shown in equation 4. The linearly decreasing inertia weight strategy has two tasks called the exploration and the exploitation, respectively. The particles in the swarm do a global search at the beginning (the exploitation). When inertia weight w is small, the particles conduct the exploration. This inertia weight control has the possibility to search and find the local solutions [14].

$$w = w_{max} - \frac{w_{max} - w_{min}}{N_{iteration}} \times iteration \tag{4}$$

where $N_{iteration}$ is the number of iterations, w is the inertia weight.

3. NEURAL NETWORK TRAINED BY IMPROVED PARTICLE SWARM OPTIMIZATION

3.1 Improved PSO algorithm

In the standard PSO, the particle may stick to the local minimum and the training will stop at this point. To overcome this limitation, an improved version of the PSO called IPSO algorithm is proposed in this paper. The velocity update function of the IPSO is modified as can be seen in equation 5.

$$\begin{aligned} v_p^{t+1} = w v_p^t + c_1 r_1 (x\_pbest_p^t - x_p^t) \\ + c_2 r_2 (x\_gbest^t - x_p^t) \\ + c_3 r_3 (\frac{1}{e^{v_p^t \times v_p^t}}) \end{aligned} \tag{5}$$

The first part of the proposed velocity update function is similar to the velocity update function of the standard PSO as presented in equation 2. To keep the particle out of the local minimum, the proposed mechanism has the second part (c_3 part). This algorithm has two phases. In the first phase, the operation of IPSO is similar to the operation of standard PSO. The second phase keeps the particle out of the local minimum. The operation of the IPSO algorithm is described as follows:

(1) When a particle has a high velocity, $e^{v_p^t \times v_p^t}$ becomes very high, c_3 part becomes 0. The velocity update function in the proposed algorithm is identical to the velocity update function in standard PSO.

(2) When the particle is near the local minimum, the velocity slows down. In this situation, the new calculated velocity starts to be influenced by the c_3 part. This additive parameter improves the exploration ability. The possibility of being trapped in the local minimum of the particle is reduced.

The proposed IPSO algorithm also uses the linear decreasing inertia weight which is presented in section 2.3.

3.2 Proposed architecture

An architecture for the implementation of the NN-PSO was designed. This architecture has two versions called the architecture for the training phase and the architecture for the testing phase.
The modules for the training of the NN-PSO are illustrated in Fig. 1. This architecture has four main components called the normalization module, the data-shuffling module, the checking module, and the NN-PSO module. In the training phase, raw data are processed in the normalization module. The output data of the normalization module become the input data of the data-shuffling module. These data are shuffled in the data-shuffling module before being processed in the NN-PSO module. The checking module is used to check the results from the NN-PSO module.

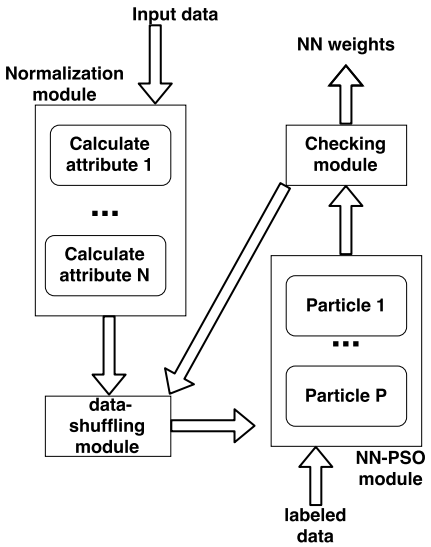

Fig. 1. Training phase of the NN-PSO

In the testing phase, raw data are also standardized in the normalization module. Finishing the normalization, the data are processed in the NN-PSO module. The weights of the NN in this module are

already calculated in the training phase. The operation of the testing phase is presented in Fig. 2.

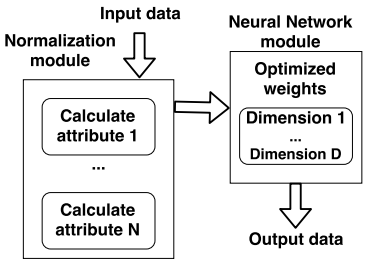

Fig. 2. Testing phase of the NN-PSO

Next subsections detail main components in the proposed architecture.

3.2.1 Normalization module. This module is the first stage of the data pre-processing. It receives raw input data and performs data normalization. The z-score is used for the standardization process. Equation 6 shows the z-score normalization [15]. This step converts the input data to a common scale for the activation function.

$$S(x) = \frac{attribute[i][j] - ave\_attribute[i]}{sd\_attribute[i]} \quad (6)$$

where $attribute[i][j]$ is the j^{th} data of the $attribute[i]$, $ave\_attribute[i]$ and $sd\_attribute[i]$ denote the mean and the standard deviation of the $attribute[i]$.
In this module, each attribute of the data is processed one-by-one. If the data have S samples, the first attribute of all S samples will be normalized in the first iteration. This processing step is repeated until all attributes are standardized.

3.2.2 Data-shuffling module. This module generates a new order of the data at any given training iteration. The operation of this module is based on the random numbers. In each iteration, two random numbers are generated N times. The data at the positions corresponding to these random numbers are swapped. For example in Fig. 3 , if 2 and $N-1$ are generated, $data[2]$ and $data[N-1]$ will be swapped.

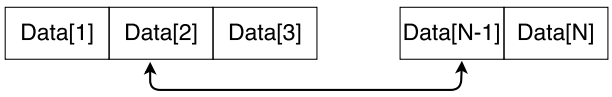

Fig. 3. Operation of the data-shuffling module

3.2.3 NN-PSO module. This is an indispensable module in the proposed architecture. In the training phase, this module implements both PSO algorithms (standard PSO, IPSO) and the NN. Fig. 4 shows the operation of the NN-PSO module in the training phase. The data from the shuffling module are sent to P particles. Each particle has D dimensions. D is also the size of the NN that can be calculated by equation 7.

$$D = (N_I+1)\times N_H+(N_H+1)\times N_H\times N_L+(N_H+1)\times N_O \quad (7)$$

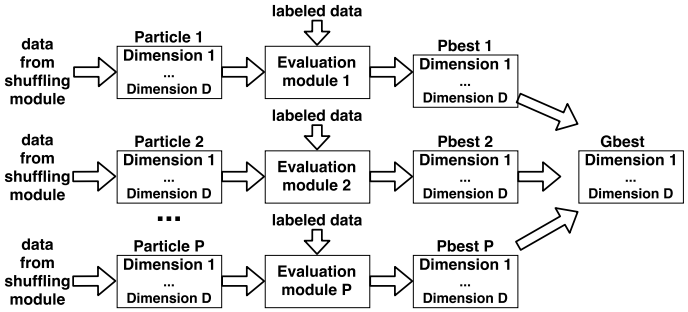

Fig. 4. Training operation of the NN-PSO module

where N_I, N_H, and N_O are the numbers of the nodes in the input layer, the hidden layer, and the output layer. N_L is the number of hidden layers as presented in Fig. 5.

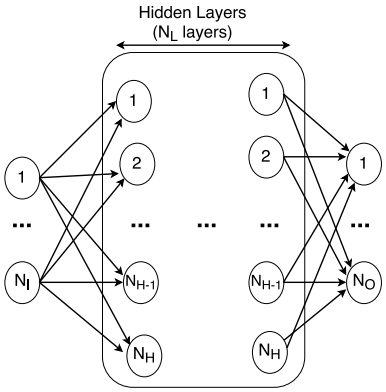

Fig. 5. Neural network

The output data of the NN in each particle are processed in the evaluation module. This module performs the mean squared error as illustrated in equation 8.

$$f_i = \frac{1}{T}\sum_{j=1}^{T}(labeled_j(k) - output_{ij}(k))^2 \quad (8)$$

where T is the number of training samples, $labeled(k)$ and $output(k)$ are the k^{th} component of the particle i in the labeled data and the output data of the NN.
From the evaluation module, the minimum learning error of each particle ($Pbest$) and the global minimum learning error ($Gbest$) of all particles are calculated. The weights of the NN correspond with the $Gbest$ will be used for the testing phase as shown in Fig. 6. The testing phase uses only the NN.

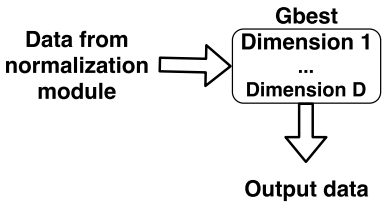

Fig. 6. The testing phase of the NN

In the NN-PSO module, the initial weights are very crucial. These initial weights cannot have the same values such as all weights are zeros. In this situation, all neurons act similar behaviors during the training process. If the required weights for the NN are unequal weights, the NN cannot learn. In addition, the different initial values may prevent the situation when the NN sticks to a local minimum.

3.2.4 Checking module. This module checks the stopping criteria in each training iteration. The stopping conditions could be the number of iterations or the final $Gbest$. If the requirements are met, the optimized NN weights are stored. These weights will be used in the testing phase. Otherwise, a new iteration will start.

4. EXPERIMENT AND DISCUSSION

The proposed architecture was implemented in the Ubuntu operating system by the C language. Based on experiments, the PSO algorithm obtained a high recognition rate and a low training error with the parameters $c_1 = 0.3$, $c_2 = 0.3$, $c_3 = 0.4$, w reduces from 0.9 to 0.0.
The experiments focused on two different aspects. The first issue evaluated the normalization module and the data-shuffling module of the proposed architecture. The second issue, the primary focus, investigated the performance of the NN trained by proposed IPSO algorithm in the proposed design when compared with the NN trained by standard PSO and the NN trained by BP.

4.1 Data pre-possessing experiment

Three different approaches for the IPSO algorithm were implemented. The first approach did not use the normalization module (No_Normalized_PSO). The data-shuffling module was not applied in the second approach (No_Shuffle_PSO). The final approach had both normalization and data-shuffling (Full_PSO). The parameters of the NN-PSO were modified randomly to investigate the operation of three algorithms in different situations, different scenarios. These parameters are the number of particles and the number of iterations.
The wine dataset was chosen from UCI machine learning database. This data set comes from the chemical analysis of Italian wines. It has three classes and thirteen attributes. In this data set, the proline attribute is much higher than other attributes. The number of data samples of this database is 178 [16].
In this experiment, 120 samples of the wine data set were selected randomly as training data, 58 remaining samples were considered as testing data. The NN used in this experiment had thirteen input nodes (corresponding to thirteen attributes), thirteen hidden nodes, three hidden layers, and three output nodes (corresponding to three classes). The parameters of the IPSO training were modified from small values to high values in three different scenarios as presented in Table I.
The approach that used both data-shuffling and normalization techniques (Full_PSO) obtained better performances than two other approaches (No_Normalized_PSO, No_Shuffle_PSO) regarding the recognition rate and the minimum learning error ($Gbest$) in all scenarios. Even with a small number of particles and iterations (10 particles, 50 iterations), the Full_PSO approach got more than 80% recognition rate.
Fig. 7 shows the reduction of the global minimum of the learning error ($Gbest$) in the first scenario. In this test, the parameters were 10 particles, and 50 iterations. The Full_PSO produced better $Gbest$ (the final $Gbest$ = 0.1848) than the No_Shuffle_PSO ($Gbest$ = 0.2936), and the No_Normalized_PSO ($Gbest$ = 0.3315).

Table I. The data pre-processing experiment of IPSO algorithm

Parameters	Approaches	$Gbest$	Recognition rate
10 particles	No_Normalized_PSO	0.3315	32.76%
50 iterations	No_Shuffle_PSO	0.2936	55.45%
	Full_PSO	0.1848	82.76%
25 particles	No_Normalized_PSO	0.2466	48.28%
100 iterations	No_Shuffle_PSO	0.0703	86.21%
	Full_PSO	0.0437	86.21%
50 particles	No_Normalized_PSO	0.1919	63.79%
1000 iterations	No_Shuffle_PSO	0.0108	93.10%
	Full_PSO	3.31×10^{-7}	96.56%

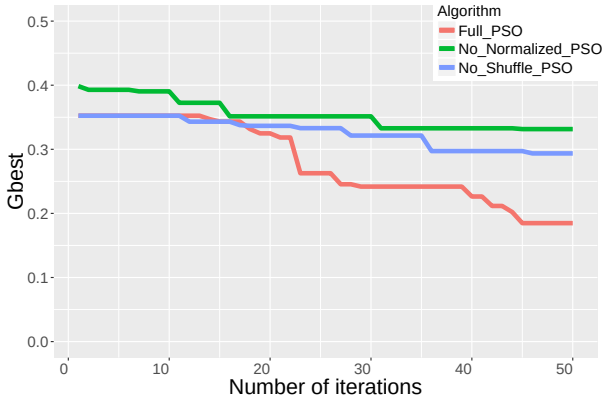

Fig. 7. 10 particles, 50 iterations

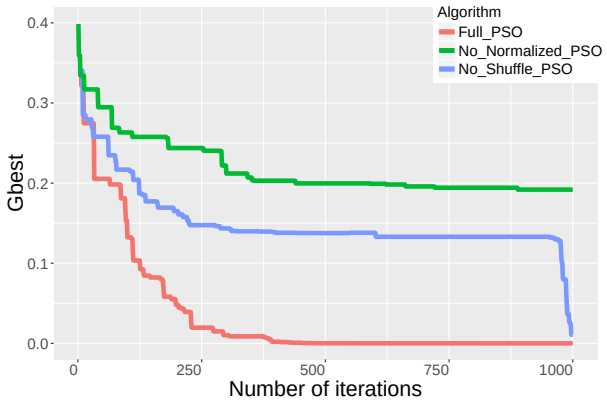

Fig. 8. 50 particles, 1000 iterations

Fig. 8 illustrates the reduction of $Gbest$ when the number of particles was 50, and the number of iterations was 100. The final $Gbest$ of the Full_PSO still reduced to the lowest value (3.31×10^{-7}). This data pre-possessing experiment confirmed the need of using both normalization module and data-shuffling module in the proposed architecture. Next section describes the IPSO experiments, the main experiments. These experiments evaluated the operation of the proposed IPSO algorithm in the proposed architecture.

4.2 IPSO experiments

These experiments investigated the NN trained by three different algorithms in the proposed architecture. The first algorithm is conventional BP algorithm. The second algorithm is standard PSO algorithm. The final algorithm is the proposed IPSO algorithm. All three algorithms were implemented in the design which had both normalization module, and data-shuffling module.
The number of iterations and the number of particles were also assigned randomly to investigate the operation of three algorithms in different situations. These experiments used three separate public recognized data sets.

4.2.1 Wine dataset . The first experiment also used the wine data set as the data pre-processing experiment. The wine data set was divided randomly into two smaller sets. Each set had 89 samples. The 2-fold cross validation was conducted. When set 1 was used as the training data, set 2 was chosen as the testing data and vice versa. The configuration of the NN in this experiment was identical to the configuration of the NN in data pre-processing experiment (thirteen input nodes, thirteen hidden nodes, three hidden layers, and three output nodes).
Table II illustrates the operations of three algorithms when set 1 was considered as the training data. In this test, the number of particles was fixed (50 particles), and the number of iterations was changed from 750 iterations to 1000 iterations. The IPSO obtained the highest recognition rate and the lowest $Gbest$ in both two scenarios (750 iterations, and 1000 iterations).

Table II. Set 1 used for training, PSO and IPSO had 50 particles

iterations	Algorithm	$Gbest$	Recognition rate
750	BP with data pre-processing	0.0615	82.02%
	PSO with data pre-processing	0.0303	87.64%
	IPSO with data pre-processing	9.8×10^{-6}	93.26%
1000	BP with data pre-processing	0.0600	83.15%
	PSO with data pre-processing	0.0077	92.14%
	IPSO with data pre-processing	2.6×10^{-9}	95.51%

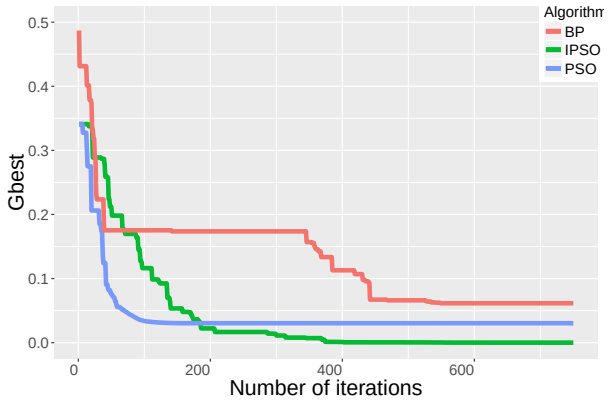

Fig. 9. Set 1 training, 750 iterations, PSO and IPSO had 50 particles

Fig. 9 and Fig 10 show the learning curve in two scenarios of this test. The $Gbests$ of the IPSO algorithm in both scenarios achieved the lowest values that indicated the performance of the IPSO algorithm. These results may be explained by the using of the proposed

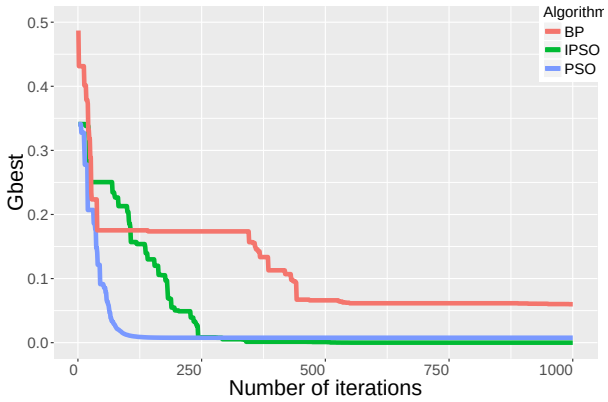

Fig. 10. Set 1 training, 1000 iterations, PSO and IPSO had 50 particles

c_3 part. This part may improve the exploration ability of the PSO algorithm. Each particle may continue to search the best location near the local minimum.
For the 2-fold cross validation, set 2 was also used as the training data and set 1 was selected as the testing data. In PSO algorithm, two important parameters are the number of iterations and the number of particles. In the previous test, the number of iterations was modified. Therefore, the changing of the particle numbers was investigated in this test. The number of iteration was kept at 1000 iterations.

Table III. NN-PSO and NN-NP when set 2 used for training, 1000 iterations

Algorithm	Particles	$Gbest$	Recognition
BP with data pre-processing	-	0.0922	84.27%
PSO with data pre-processing	55	0.0156	87.64%
	70	0.0071	89.89%
IPSO with data pre-processing	55	3.9×10^{-7}	91.01%
	70	1.8×10^{-7}	95.51%

Table III presents the experimental results when the number of particles in the PSO, IPSO algorithms was changed from 55 particles to 70 particles. In both configurations of the particle numbers, the IPSO algorithm got better performance regarding the recognition rate, and the $Gbest$ than the standard PSO and the BP algorithm.

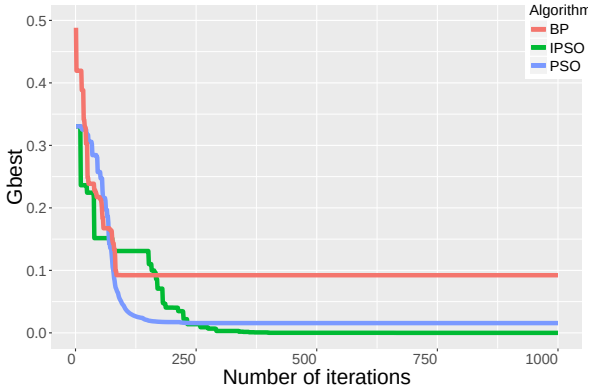

Fig. 11. Set 2 training, 55 particles, 1000 iterations

Fig. 11 shows the reduction of $Gbest$ when set 2 was used as the training data, set 1 was considered as the testing data. The parameters for PSO, IPSO were 55 particles, 1000 iterations. The $Gbest$ of the IPSO, declined to the lowest values, confirmed the performance of the proposed IPSO.

4.2.2 Statlog (heart) data set . This is the heart disease database which has thirteen attributes, and two classes called the presence of heart disease and the absence of heart disease [16]. The serum cholesterol attribute in this database has a higher value than other attributes. The purpose of the experiment with this data set is to investigate the operation of the NN-PSO system with different databases. The settings of the NN were thirteen input nodes, two output nodes, thirteen hidden nodes, and three hidden layers.
In this experiment, 100 samples were chosen randomly as the training data; another 50 samples were selected randomly as the testing data. The number of particles was 100 particles; the number of iterations was 1000 iterations.

Table IV. Stalog data set

Algorithm	$Gbest$	Recognition rate
BP with data pre-processing	0.1279	84.00%
PSO with data pre-processing	0.0637	86.00%
IPSO with data pre-processing	0.0186	92.00%

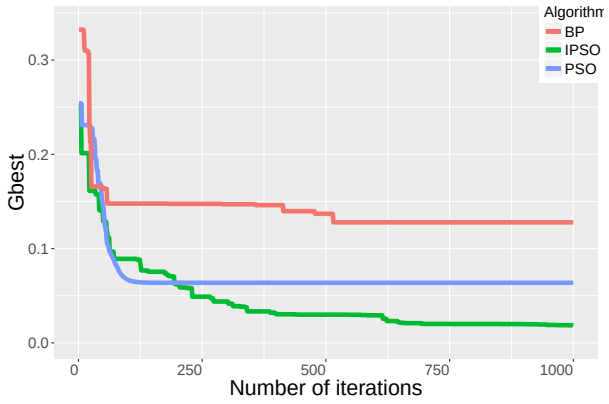

Fig. 12. Statlog (heart) data set, 100 particles, 1000 iterations

Table IV and Fig. 12 present the results of this experiment. Experimental results showed that the proposed IPSO achieved the best performance among the three algorithms (IPSO, PSO, and BP) concerning the recognition rate and the global minimum of the learning error.

4.2.3 Mesothelioma dataset. In wine experiment and statlog experiment, the data sets have only thirteen features. This experiment investigated the operation of the NN trained by proposed IPSO algorithm with a bigger data set. The mesothelioma dataset was used in this experiment. This data set comes from the mesotheliomna disease diagnosis of 324 patients. Each patient data has 34 features [17]. To test on this database, the configurations of the NN were 34 input nodes, 34 hidden nodes, three hidden layers, and two output nodes.

From 324 patient data, 280 samples were considered as the training data; 80 remaining data samples were the testing data in this experiment.

Table V. Mesothelioma dataset

Parameters	Algorithm	$Gbest$	Recognition
40	BP with data pre-processing	0.0615	82.02%
500	PSO with data pre-processing	0.0303	87.64%
	IPSO with data pre-processing	9.8×10^{-6}	93.26%
40	BP with data pre-processing	0.0600	83.15%
4000	PSO with data pre-processing	0.0077	92.14%
	IPSO with data pre-processing	2.6×10^{-9}	95.51%

Table V shows the results of the mesothelioma experiment. In the first scenario, the parameters were 40 particles and 500 iterations. The NN trained by IPSO obtained the highest recognition rate (93.26%). The recognition rate of the NN trained by PSO was 87.64%, and the recognition rate of the NN trained by BP was 82.02%. In the next scenario, the number of iterations was increased to 4000 iterations. The recognition rates of all three algorithms were improved. The IPSO still got the highest recognition rate (95.51%).

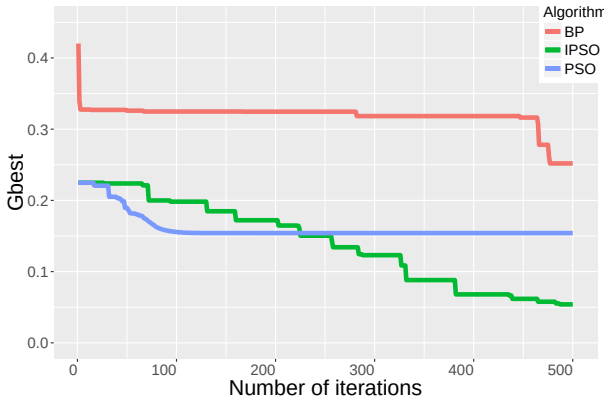

Fig. 13. Mesothelioma data set, 40 particles, 500 iterations

Fig. 13 shows the reduction of $Gbest$, the global minimum learning error, in one scenario of this experiment. In this scenario, the number of particles was 40 particles, and the number of iterations was 4000 iterations. Among three algorithms, IPSO obtained the lowest $Gbest$ (9.8×10^{-6}).
The results came from the experiments with three different databases confirmed the better performance of the proposed IPSO concerning the recognition rate and the training error.

4.3 Two hidden layers experiments

In previous experiments presented in section 4.1 and section 4.2, the three hidden layers NN was tested. It is beneficial to investigate the operation of the proposed IPSO with different sizes of the NN. The experiment presented in this section was conducted with two hidden layers NN. Other configurations of the NN were thirteen input nodes, thirteen hidden nodes, and three output nodes. The wine data set was used in this experiment. The wine samples were divided randomly into three different subsets. Set 1 had 60 instances, set 2 had 60 instances and set 3 had 58 samples. The 3-fold cross

validation was done with three different algorithms (IPSO, PSO, and BP). Each algorithm had three different approaches (without the normalization module, without the data-shuffling module, with both the normalization module and the data-shuffling module). The number of particles $P = 50$, the number of iterations $I = 1000$.

Table VI. Two hidden layers NN

Algorithms	Approaches	Recognition
BP	No_Normalized_BP	26.9767%
	No_Shuffle_BP	90.4233%
	Full_BP	90.4233%
PSO	No_Normalized_PSO	64.5966%
	No_Shuffle_PSO	88.1484%
	Full_PSO	92.6437%
IPSO	No_Normalized_PSO	70.23%
	No_Shuffle_PSO	93.8317%
	Full_PSO	95.4993%

Table VI shows the results of the 3-fold cross validation in three different algorithms with two hidden layers NN. In all approaches, the proposed IPSO got the highest recognition rate. These results also demonstrated the efficiency of data pre-processing modules which may improve the recognition rate.

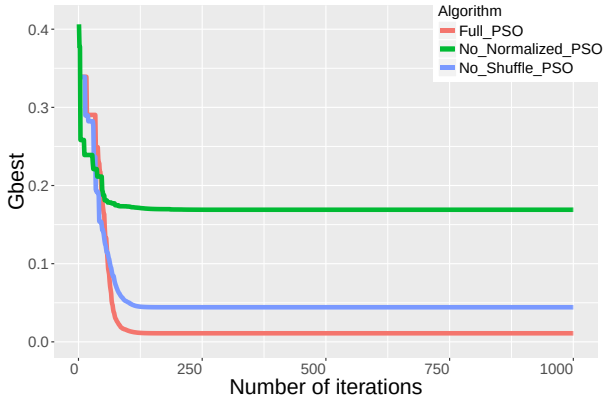

Fig. 14. Set 1 and set 3 training, set 2 testing, PSO algorithm

Fig. 14 shows the results of PSO algorithm when set 1 and set 3 were used as the training data, set 2 was considered as the testing data. The final $Gbests$ in Full_PSO, No_Shuffle_PSO, and No_Normalized_PSO were 0.011, 0.044, and 0.169, respectively.
Fig. 15 describes the reduction of the $Gbest$ in IPSO algorithm when set 1 and set 2 were used as training data, set 3 was used as testing data. The final $Gbest$ of the No_Normalized_PSO was 0.1838. On the other hand, the final $Gbests$ of Full_PSO and No_Shuffle_PSO were small. Comparing these two algorithms, the $Gbest$ of Full_PSO got a smaller value (7.36×10^{-9}) than the $Gbest$ of No_Shuffle_PSO (3.93×10^{-8}).
Experimental results presented in Table VI, Fig. 14, and Fig. 15 suggested that the data pre-processing modules (normalization, data-shuffling) may improve the recognition rate of all three algorithms (IPSO, PSO, and BP). The proposed IPSO also obtained the best performance among the three algorithms concerning the recognition rate and the global minimum of the learning error.

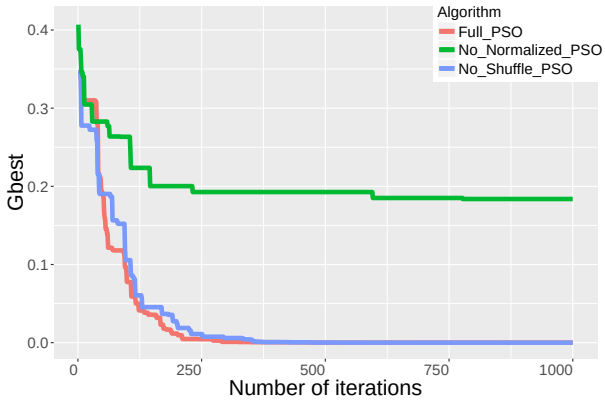

Fig. 15. Set 1 and set 2 training, set 3 testing, IPSO algorithm

4.4 Discussion

In the light of the evidence, experiments confirmed that the proposed IPSO achieved the highest performance among three algorithms (IPSO, PSO, and BP) with different settings of the PSO algorithms (number of iterations, number of particles), different databases (wine, statlog, and mesothelioma data sets), different size of the NN (two hidden layers, three hidden layers). The c_3 part in the IPSO algorithm improved the exploration ability of the standard PSO algorithm. This mechanism helps the algorithm continue to search in the area that nears the local minimum. Even when the standard PSO gets stuck in the local minimum, the IPSO may continue to search the optimal solutions.
Considering the experimental results, both two PSO algorithms (standard PSO, IPSO) got better performance than the BP algorithm.
In the experiments, the normalization module and the data-shuffling module also increased the recognition rate and reduced the learning error of the NN. These results confirmed the need of the data-preprocessing not only with the NN trained by BP algorithm but also with the NN trained by PSO (standard PSO, IPSO). Once the input data has been sent to the NN-PSO or NN-BP, this data must be pre-processed. Otherwise, the NN will not produce accurate results.
Experimental results also demonstrated that the NN trained by No_Shuffle_PSO got higher performance than the NN trained by No_Normalized_PSO regarding the learning error and the recognition rate. These results suggested that the normalization module had a more significant role in the data-preprocessing than the data-shuffling module in the NN-PSO architecture.

5. CONCLUSION

This paper proposes the improved version of PSO algorithm called IPSO algorithm. Experimental results demonstrated the advantages of the proposed IPSO regarding the recognition rate and the learning error when compared with the standard PSO algorithm and the BP algorithm.
The architecture for the NN trained by PSO (standard PSO, ISPO) was also successfully implemented in this paper. The operation of NN trained by three different algorithms (IPSO, PSO, BP) in the proposed architecture is evaluated in this paper. Each algorithm has three different approaches. The first approach does not have the data-shuffling module. The second approach does not use the normalization module. The third approach consists not only the

data-shuffling mechanism but also the normalization mechanism. Experimental results showed that the third approach had the highest recognition rate and the lowest global learning error. Considering the results, the normalization method and data shuffling method are the essential techniques for the training of the NN.

The future research will investigate the role of the data pre-processing module with more complex data sets and bigger NNs. Another possible avenue for the research is to investigate other methods for the data pre-processing such as noise reduction, outlier reduction, or remove the redundancy. Further research about the proposed IPSO algorithm will be conducted.

The future scope of this research is the development of a useful architecture for the daily life. Therefore, the proposed architecture with IPSO algorithm and the data pre-processing modules needs to be confirmed with real-life applications. In this situation, the data come from the daily life situations such as the credit card approval of the credit card company or the clothes selection of the stores will be fed to the proposed architecture.

6. REFERENCES

[1] S. Haykin, *Neural networks and learning machines*, 3rd edn, Prentice Hall, 2008

[2] R. H. Nielsen, Theory of the backpropagation neural network, *In processing of the international conference on neural networks*, pp. 693-605, 1989

[3] R. Rojas, *Neural networks - a systematic introduction*, Springer-Verlag, 1996

[4] J. R. Zhang, J. Zhang, T. M. Lok. M. R. Lyu, A hybrid particle swarm optimizationback-propagation algorithm for feedforward neural network training, *Applied mathematics and computation*, vol. 185, pp. 10261037, 2007

[5] Z.A. Bashir, M.E. El-Hawary, Applying Wavelets to Short-Term Load Forecasting Using PSO-Based Neural Networks, *IEEE transactions on power systems*, vol. 46, pp. 268-275, 2016

[6] A. Suresh, K. V. Harish, N. Radhika, Particle Swarm Optimization over Back Propagation Neural Network for Length of Stay Prediction, *In processing of the international conference on information and communication technologies*, vol. 24, no.1, pp. 20-27, 2009

[7] V. G. Gudise, G. K. Venayagamoorthy, Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks, *In processing of 2003 IEEE swarm intelligence symposium*, pp. 110-117, 2003

[8] M. T. Das, L. C. Dulger, Signature verification (SV) toolbox: Application of PSO-NN, *Engineering applications of artificial intelligence*, vol. 22, issue 4-5, pp. 688-694, 2009

[9] R. Mendes, et al., Particle swarms for feedforward neural network training, *In processing of the IEEE international joint conference on neural networks*, vol.2, pp.1895-1899, 2002

[10] K. W. Chau, Application of a PSO-based neural network in analysis of outcomes of construction claims. *Automation in construction*, vol. 16, no. 5, 642-646, 2007

[11] G. Montavon, G. B. Orr, K. R. Muller *Neural networks: tricks of the trade*, 2nd edn, Springer, 2012

[12] J. Kennedy, R. Eberhart, Particle swarm optimization, *In processing of the IEEE international conference on neural networks*, vol. 4, pp.1942-1948, 1995

[13] R. Eberhart, Y. Shi, Particle swarm optimization: developments, applications and resources, *In processing of the 2001 IEEE international conference on congress on evolutionary computation*, vol. 1, pp. 81-86, 2001

[14] Y. Shi and R. Eberhart, Empirical study of particle swarm optimization, *In processing of international conference on evolutionary computation*, pp. 1945-1950, 1999

[15] J. Han, M. Kamber, J. Pei, *Data mining: concepts and techniques*, 3rd edn, Morgan Kaufmann, 2011

[16] M. Lichman, UCI Machine Learning Repository, http://archive.ics.uci.edu/ml, accessed Sep. 03 2016

[17] E. Orhan, A. C. Tanrikulu, A. Abakay, F. Temurtasa, An approach based on probabilistic neural network for diagnosis of Mesotheliomas disease, *Comput electr eng*, vol. 38, issue 1, pp. 75-81, 2012